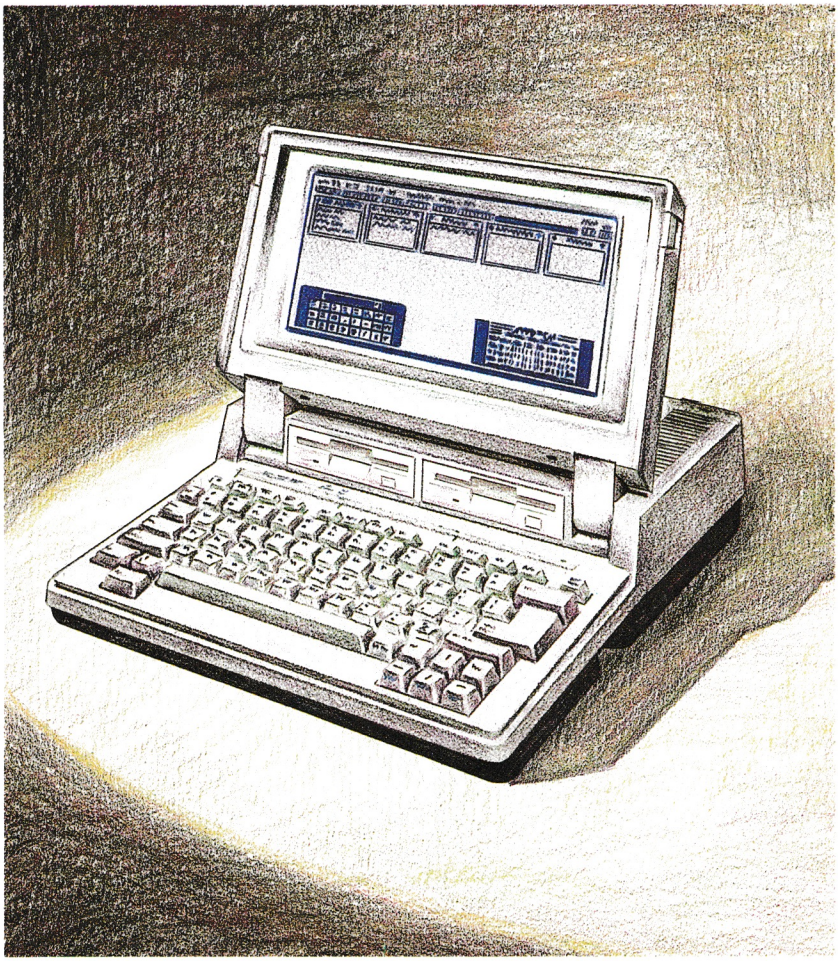
TANDY 1400 FD/HD

MS-DOS/GW-BASIC
Quick Reference Guide



TANDY®

Tandy 1400 FD/HD Quick Reference Guide
© 1989 Tandy Corporation.
All Rights Reserved.

Reproduction or use of any portion of this manual, without express written permission from Tandy Corporation and/or its licensor, is prohibited. While reasonable efforts have been mede in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors in or omissions from this manual, or from the use of the information contained herein.

GW is a trademark of Microsoft Corporation.

10 9 8 7 6 5 4 3 2

MS-DOS

Quick Reference

Loading MS-DOS 2
MS-DOS Commands 3
Special Keys 33
Edlin Commands 36
Edlin Editing Keys 40
Debug Command Parameters 41
Debug Commands 43
GW-BASIC Quick Reference 49
Loading GW-BASIC 50
GW-BASIC Commands and Statements 52
GW-BASIC Function Key Settings 102
Typing Keywords Using the ALT Key 102
Exponential Notation and Numeric Precision Characters 103
Operator Precedence 103
Video Modes 104
Error Codes and Messages 105

Loading MS-DOS

1. Turn on your computer.
2. To start from floppy diskette, insert the MS-DOS/BASIC diskette into Drive A.

   To start from an initialized hard disk (Drive C), do not insert a diskette in Drive A. MS-DOS automatically starts up from Drive C.
3. When the date prompt appears, enter the current date in the *mm-dd-yy* format. (For example, type `9-7-89` or `09-07-1989`, and press ENTER for September 7, 1989).
4. When the time prompt appears, either verify the displayed time by pressing ENTER, or change the time by entering it in the *hh:mm:ss.cc*, 24-hour format. (For example, type `14:30`, and press ENTER for 2:30 p.m.)

   If you are booting from floppy diskette, MS-DOS displays the system prompt:

   ```
   A>
   ```

   If you are booting from hard disk, MS-DOS displays the system prompt:

   ```
   C>
   ```

   Your computer is now ready for use.

MS-DOS Commands

This chapter contains an alphabetical reference to all of the MS-DOS commands. At the beginning of each command description is a syntax line that gives the format for entering the command. Each syntax line observes the following notations:

bold	indicates information that you type exactly as it appears, such as command names, switches, and options.
`screen type`	indicates words and characters you type to perform a specified function.
lowercase italics	represent variable words, letters, characters, or values.
[ ] (square brackets)	indicate optional parameters.
... (ellipsis)	indicates that you can repeat a parameter, along with any applicable punctuation.
| (vertical bar)	indicates an either/or situation.
SMALL CAPS BOLD	indicates a key combination, such as **CTRL-Z**. Press the second key while holding down the first key.

Additionally, the commands in this section are labeled as either internal or external according to the following definitions:

Internal	these commands load into your computer's memory on startup. Your computer does not need to access a disk to use them.
External	these commands are on disk. Your computer must access the disk and directory where these commands reside in order to use them.

append [;] [*pathname* [;*pathname*]...] [/e][/x]

(External) Sets a data file path, which tells MS-DOS the drives and directories in which to search for data files. Omit all parameters to display the current data path.

/e	causes **append** to store appended directories in the MS-DOS environment.
/x	extends the search path to include all subdirectories within the specified directory.
;	sets the NUL data path; MS-DOS searches only the current directory.

```
append B:\sales\region 1;a:
```

assign [*drive1* = *drive2* ...]

(External) Reassigns the drive letter *drive1* to *drive2*. Use this command to run application programs from drives other than those for which they were written.

drive1	the original target drive.
drive 2	the substitute target drive.

```
assign a=c b=c
```

attrib [+r|-r] [+a|-a] *pathname* [/s]

(External) (1) Sets the read-only and archive attributes of the file specified by pathname, or (2)displays the attributes if you omit the optional parameters.

+r	sets the read-only mode.
-r	disables the read-only mode.
+a	sets the archive attribute.
-a	clears the archive attribute.
/s	causes **attrib** to apply to all subdirectories as well as the specified directory.

```
attrib +r b:\mydir\myfile.txt
attrib *.*
```

autofmt [/b]

(External) Initializes a hard disk for use. If the hard disk is Drive C, **Autofmt** installs the system files to make it the boot disk. If the hard disk is larger than 32 megabytes, Autofmt partitions it as necessary.

/b causes **autofmt** to prompt you for bad sector information.

```
autofmt
```

backup [*source pathname*] [*target drive:*][**/s**] [**/m**] [**/a**] [**/f**][**/d:***mm/dd/*yy] [**/t:***hh:mmx*][**/l:***filename*]

(External) Backs up one or more files from one disk to another formatted disk. Backup can copy between disks of different media, for example from hard disk drives to floppy disk drives. It can also copy from one diskette to another, even if the diskettes have a different number of sides and sectors.

source pathname specifies the files to back up. It can be an entire drive, a directory name, or a filename.

target drive: the drive to receive the files. If the target is a diskette drive, **backup** places the files in the root directory.

If it is a hard disk drive, **backup** places the files in a subdirectory called Backup.

/s copies all files in the specified directory and in the directories below it.

/m copies only those files modified since the last backup.

/a adds the files to be backed up to those already on the target disk, instead of erasing the existing files.

/f formats the target disk if it is not already formatted.

/d:*mm/dd/yy* copies only those files created on or after the specified date.

/t:*hh:mmx* backs up only those files modified on or after the specified time. The time must be less than 13:00. *x* is either a (for a.m.) or p (for p.m.).

/l:*filename* creates a backup log entry in the specified file or – if you omit *filename* – in a file called Backup.log in the root directory of the files being backed up.

```
backup c:store\sales.dat a: /a
```

break [on|off]

(Internal) Turns the **CTRL-C** check on or off. Displays the current setting of **CTRL-C** if you omit the parameters.

```
break off
```

cache [*buffer size*]
cache *[drive:*][**/8**][**/9**][/**c**][/**g**][/**o**][/**r**][/**s**][/**t**:*seconds*][/**w**]

(External) Establishes a RAM buffer for storing data coming from or going to a disk.

buffer size the size of the buffer in kilobytes. Default = 30K.

After you create the cache buffer, you can again issue the **cache** command with the following options:

drive: the drive letter of the drive for which the following parameters are effective. Default = current drive.

/8 sets for a diskette formatted 8 sectors per track.

/9 sets for a diskette formatted 9 sectors per track.

/c clears the accumulated statistics. (See **/s**.)

/g displays the current cache status.

/o toggles **cache** on and off.

/r resets all data in the buffer.

/s displays the cache statistics.

/t:*seconds* sets the time in seconds that data stored in the buffer from diskette drives remains valid.

/w toggles disk saves to the buffer on and off. This is only valid for buffers smaller than 100K.

```
cache
cache a: /t:1 /w
```

chcp [*nnnn*]

(Internal) Displays or changes the current code page for the command processor.

nnnn the code page to display or select. **Chcp** accepts one of the two prepared system code pages. An error message appears if you select a code page that is not prepared. If you type the **chcp** command without a code page, it displays the active code page and the prepared code page.

The following are valid code pages:

437	United States
850	Multilingual
860	Portuguese
863	French-Canadian
865	Nordic

```
chcp
chcp 863
```

chdir [*pathname*]
cd [*pathname*]

(Internal) Changes the current or home directory of the specified drive to the directory specified by *pathname*. Displays the pathname of your current directory if you omit *pathname*.

```
chdir \bin\user
chdir b:\user
```

chkdsk [*pathname*] [**/f**] [**/v**]

(External) Checks the MS-DOS disk in the current or specified drive for errors. You can redirect the **chkdsk** output to a file by adding >*pathname2* to the end of the command.

pathname	specifies either an entire drive or an individual file. If you specify a file, **chkdsk** displays information about both the drive and the file.
/f	fixes errors (if possible) and updates the disk. (Do not redirect the output from **chkdsk** if you use **/f**).
/v	displays messages and error details while **chkdsk** is running.

```
chkdsk a:\sales\jrsales>>b:\sales\jrerrs
```

cls

(Internal) Clears the screen.

```
cls
```

command [*pathname*] [*device*] [**/e:***size*] [**/p**] [**/c** *string*]

(External) Starts a new command processor.

pathname specifies the drive and directories in which the command processor is to look for the Command.com file if it needs to reload the transient portion of the file into memory.

device specifies a different device for input and output. It can be:

- aux — to specify an auxiliary device, usually RS232 Serial Port 1.
- com1 — to specify RS232 Serial Port 1.
- com2 — to specify RS232 Serial Port 2.
- con — to specify the console (keyboard input, screen output).

/e:size specifies the environment size, in bytes. *Size* is in the range 160-32768. The default is 160.

/p tells the command processor not to exit to a higher level.

/c string tells the command processor first to execute the command or commands specified by *string*, then to return. The **/c** switch is valid only as the last parameter.

```
command /c chkdsk b:
```

comp [*drive***:**][*pathname1*][*drive***:**][*pathname2*]

(External) Compares the contents of two files or sets of files.

```
comp c:*.asm b:*.bak
```

copy *source pathname* [*target pathname*] **[/a] [/b] [/v]**

(Internal) Copies one or more files to the same directory as the source (giving them different filenames) or to another directory (giving them the same or different filenames). To leave the filename the same, omit the filename from *target pathname*. If you omit **/a** and **/b**, **copy** uses **/b**.

/a Source file: treats the file as an ASCII file (text or data file). Target file: adds an EOF character to the end of the file.

/b Source file: treats the file as a binary file (program file). Target file: does not add an EOF character to the end of the file.

/v verifies the sectors written to disk.

```
copy memos.txt /a b:corr.txt
```

copy *target pathname* + *source pathname1*[+*source pathname2*...] **[/a] [/b] [/v]**

(Internal) Adds one or more files to the end of the existing file specified by target pathname. If you omit **/a** and **/b**, **copy** uses **/a**.

/a Source file: treats the file as an ASCII file (text or data file). Target file: adds an EOF character to the end of the file.

/b Source file: treats the file as a binary file (program file). Target file: does not add an EOF character to the end of the file.

/v verifies the sectors written to disk.

```
copy b:read.dat + write.dat + print.dat
```

copy *source pathname1* [*+source pathname2...*]*target pathname* **[/a]** **[/b]** **[/v]**

(Internal) Combines any number of source files into a new file, specified by target pathname. If you omit **/a** and **/b**, **copy** uses **/a**.

/a	Source file: treats the file as an ASCII file (text or data file). Target file: adds an EOF character to the end of the file.
/b	Source file: treats the file as a binary (program) file. Target file: does not add an EOF character to the end of the file.
/v	verifies the sectors written to disk.

```
copy b:memos.txt + b:letters.txt b:corr.txt
```

cpanel

(External) Allows the user to adjust the sensitivity of his/her mouse. In order to invoke **cpanel** the user must enter the following key combination:

CTRL - ALT - (Left button on mouse)

Note: Microsoft mouse driver must be previously loaded.

```
cpanel
```

ctty *device*

(Internal) Changes the I/O *device* to the *device* specified.

device can be any of the following:

aux	to specify RS232 Serial Port 1.
com1	to specify RS232 Serial Port 1.
com2	to specify RS232 Serial Port 2.
con	to specify the console.

```
ctty aux
```

date [*mm/dd/yyyy*]

(Internal) Enters or changes the system date. Displays the current date if you omit the date parameter.

mm/dd/yyyy specifies the month, day, and year to set as the date.

```
date 11/15/1987
date 11/15/87
```

dc [**/c**][**/d**][**/q**][*pathname*]

(External) Compresses the specified file or returns it to its original state.

/c compresses the specified file.

/d decompresses the specified file.

/q displays a message indicating whether or not the specified file is compressed.

```
dc /c myfile
dc /d a:\region1\sales
```

del

(Internal) See **erase.**

dir [*pathname*][**/p**][**/w**]

(Internal) Displays information about: (1) files in the current directory, (2) files in the directory specified by *pathname*, or (3) the one file specified by *pathname*.

/p selects the "page" mode.

/w selects a wide display.

```
dir b:
dir \user\*.bat/p
```

diskcomp *[drive1*:][*drive2*:][**/1**][**/8**]

(External) Compares the contents of two diskettes.

drive1:	the drive containing the source diskette.
drive2:	the drive containing the target diskette.
/1	compares only the first side of two double-sided diskettes. If you omit **/1**, **diskcomp** compares both sides.
/8	compares only the first 8 sectors of each track. If you omit **/8**, **diskcomp** automatically compares either 9 or 15 sectors, according to the format of the two diskettes.

```
diskcomp a: b:
```

diskcopy [*source drive*:][*target drive*:]

(External) Copies the contents of the diskette in the *source drive* to the diskette in the *target drive*. The target diskette must be of the same density as the source diskette. If the target is unformatted or is formatted differently than the source diskette, **diskcopy** formats it with the same format as the source diskette. Use **diskcomp** to verify the accuracy of the duplication.

```
diskcopy
diskcopy a: b:
```

diskopt

(External) Optimizes the file storage on the current disk by rearranging its files into a contiguous format. Do not use **diskopt** with copy-protected programs.

```
diskopt
```

disktype [*drive*:]

(External) Displays information about the size and capacity of the indicated disk. For a floppy diskette, **disktype** displays the number of sides, tracks, and sectors per track. For a hard disk, it displays the number of heads, cylinders, and sectors per track. Omit *drive*: to display information about the current drive.

```
disktype c:
```

echo [on][off][*message***]**

(Internal) Turns the batch echo feature on or off, displays the specified message, or – if you omit all parameters – displays the current setting of **echo.**

```
echo off
echo Insert disk
```

erase [*pathname*]
del [*pathname*]

(Internal) Erases (deletes) one or more files from the current or specified directory. Omitting *pathname* erases all files in the specified directory.

```
erase \bin\user\jd\jd.txt
del b:\sales\joe
```

exit

(Internal) Exits the command processor and returns to a previous level, if one exists.

```
exit
```

fastopen [*drive*:[=*nnn*][...]]

(External) Decreases the amount of time needed to open frequently used hard disk files and directories. Each time you open a file or directory, **fastopen** records its name and location. Then, if you reopen the file or directory, your system immediately knows where to find it.

nnn — the number of files on the specified hard disk that you want **fastopen** to track (in the range 10-999). The default is 10.

```
fastopen c:=100
fastopen c:
```

fdisk

(External) Creates, changes, deletes, or displays hard disk partitions.

```
fdisk
```

find [/v][/c][/n] *"string"* [*pathname...*]

(External) Searches for the specified string of text in one or more files, specified by *pathname(s)*. Searches for *string* among the lines from the current console input device if you omit *pathname*.

/v — displays all lines that do not contain *string*.

/c — displays only the number of lines in each file that contain *string*.

/n — displays each line's relative line number in that file; do not use with **/c**.

```
find /n "mispell" doc.txt document.txt
```

for *%c* **in** *(set)* **do** *command* **(regular)**
for *%%c* **in** *(set)* **do** *command* **(batch file)**

(Internal) Executes the specified command for each item in the set.

set	a list of items, separated by spaces, or one wild card item.
c	can be any one-character variable except 0-9. If you include %c or %%c at the end of the command, MS-DOS sequentially substitutes each member of set in *command*. If you do not include it, MS-DOS executes *command* the appropriate number of times but does not substitute the members of *set*.

```
for %f in (taxfile autofile homefile) do
del %f
```

format [*drive*:][**/1**][**/8**][/v][/s]

(External) Prepares the disk in the specified drive. When formatting a hard disk, first use **hsect** and **fdisk** to initialize it.

drive:	the drive containing the disk to format.
/1	formats a diskette for single-sided use (The default is double-sided.)
/8	formats a diskette for 8 sectors per track.
/v	prompts for a volume label.
/s	copies the system files to the disk.
/b	leaves space on the disk to install the MS-DOS system files.

```
format
format b: /s/v
```

goto *label*

(Internal) Used in a batch file to transfer execution to the line following the line that contains *:label*.

label is a character string.

```
:g
rem looping...
goto g
```

graftable [*xxx*]

(External) Loads additional character data into a table in memory for use with a color or graphics adapter.

xxx is a code page identification number. The following are valid page codes:

/sta	request status only
437	United States
860	Portuguese
863	French-Canadian
865	Nordic

```
graftable 437
```

graphics *ptype* **[/r][/b][/cr][/lf]**

(External) Enables you to reproduce a graphics screen in color on the Tandy CGP-220 printer or in shades of gray on other printers. To reproduce the screen, press **SHIFT-PRINTSCRN.**

ptype is one of these printer types:

cgp220	the Tandy CGP-220.
dmp110	the Tandy DMP-110.
pcmode	a Tandy printer with a dip switch set for the PC mode or other PC compatible printers.
tmode	a Tandy printer with the dip switch set for the Tandy mode.
standard	any other Tandy printer.

mkdir *pathname*
md *pathname*

(Internal) Creates a directory.

pathname tells MS-DOS the directory under which to create the new directory and specifies the name to give the new directory.

```
mkdir \user
md b:\letters
```

mode [*display*]

(External) Sets the video display mode.

display the color and line width in characters (40 or 80). Display can be:

40	for 40-character display
80	for 80-character display
bw40	for monochrome 40-character display
bw80	for monochrome 80-character display
co40	for color 40-character display
co80	for color 80-character display

```
mode co80
```

mode lpt1[**:**][*chars*][**,**[*lines*][**,p**]]

(External) Sets the printer parameters.

chars specifies the characters per line (80 or 132).

lines specifies vertical spacing (6 or 8 lines per inch).

p specifies continuous retry if a printer timeout error occurs.

```
mode lpt1: 80
```

mode lfoff | lfon

(External) Turns printer linefeed off or on. Before using this command, you must load the Lpdrvr.sys device driver or execute the **lf** command.

```
mode lfoff
```

mode com *number*: [*baud*][*parity*][*databits*][*stopbits*][**p**]

(External) Sets RS232 communication parameters.

number	the RS232 serial port, either 1 or 2.
baud	can be 110, 150, 300, 600, 1200, 2400, 4800, or 9600. The default is 300 baud.
parity	can be n (no parity), o (odd parity), or e (even parity). The default is e.
databits	can be either 7 or 8. The default is 7.
stopbits	can be either 1 or 2. The default is 1.
p	tells the printer driver to continuously try to output on timeout errors.

```
mode com1:1200 N 8 1 p
```

mode *device* **codepage prepare** = ((*cpages*)[*drive*:[*path*]] *filename*)
mode *device* **codepage select**=*yyy*
mode *device* **codepage refresh**
mode *device* **codepage [/status]**

(External) Sets display code pages for parallel printer or console screen devices.

device	specifies the device to support code page switching. Valid device names are con, prn, and lpt1.
yyy	specifies a code page. Valid code pages are 437, 850, 860, 863, and 865.
filename	identifies the name of the code page information (.cpi) file MS-DOS should use to prepare a code page for the specified device.

cpages	specifies a list of valid code pages. (See *yyy*.)
prepare	tells MS-DOS to prepare code pages for a given device.
select	specifies the code page you want to use with a device.
refresh	reinstates code pages that are lost due to an error.
/status	displays the current code pages prepared and/or selected for a device.

```
mode con codepage
```

more

(External) Reads from standard input and displays one screen of information at a time, with the message `--MORE--` at the bottom. Press the space bar to see the next screen.

patch *pathname,address,data1,data2*

(External) Lets you make minor corrections in any disk file.

pathname	is the file you want to change.
address	is the starting address of the data to be changed (in hexadecimal).
data1	is the list of the hexadecimal values to be changed.
data2	is a list of the hexadecimal data values to replace data1.

```
patch b:filex.exe, 1A23, 4462A3,4360B2
```

path [[;][*pathname*][;*pathname*...]]

(Internal) Sets a command path, which tells MS-DOS the directories or drive in which to search for external commands. Displays the current path setting if you omit pathname.

pathname — specifies a directory or an entire drive.

; — used without specifying a *pathname*, sets the path to "no path", which causes MS-DOS to search only the current directory.

```
path \bin\user\joe;b:\bin\user\joe
```

pause [*message*]

(Internal) Suspends execution of the batch file.

message — a message to be displayed when execution pauses.

```
pause Insert diskette.
```

print [*pathname* [**/d**:*device*][**/b**:*size*][**/u**:*value*][**/m**:*value*] [**/s:***timeslice*][**/q**:*value*][**/c**][**/p**]...][**/t**]

(External) Puts files in the print queue for background printing.

pathname — specifies the file(s) to print.

/d:*device* — specifies the print device. Lpt1 is the default.

/b:*size* — sets the size (in bytes) of the internal buffer. The default is 512 bytes.

/u:*value* — specifies the number of clock ticks **print** will wait for a printer. Default = 1.

/m:*value* specifies the number of clock ticks (1-255) **print** can take to print a character.

/s:*timeslice* specifies the interval of time to be used by the MS-DOS scheduler for the print command.

/q:*value* selects the number of files (4-32) allowed in the print queue. The default is 10.

/c deletes (cancels) from the print queue the file that immediately precedes and all files that follow **/c** in the command line.

/p turns on the print mode and adds the preceding and all following filenames to the print queue.

/t deletes (terminates) all files in the print queue. (Do not use **/t** with *pathname*.)

```
print /t
print temp1.tst /c temp2.tst /p temp3.tst
```

prompt [*text*]

(Internal) Changes the system prompt to text. Sets the prompt to the current drive specification if you omit *text*.

text a string of characters that generates the prompt. Precede special characters with a dollar sign. Special characters are:

Specify this:	To set this:
$	$
t	the current time
d	the current date
p	the current directory
v	the version number
n	the current drive
g	>
l	<
b	|
q	=
e	an escape code
h	a backspace

```
prompt $p$g
```

rcrypt *pathname1* [*pathname2*]

(External) Changes the format of a file so that the contents are meaningless without the encryption key. The encryption key is 0-8 characters that **rcrypt** uses to alter the file contents.

pathname1 — the pathname of the file to encrypt.

pathname2 — the pathname of the file to receive the encrypted file. If you omit *pathname2*, **rcrypt** uses the standard output device.

```
rcrypt myfile b:\secret.fil
```

recover [*drive*: | *pathname*]

(External) Recovers the file (specified by *pathname*) that contains bad sectors, or recovers all files on a disk (specified by *drive*:) that contains bad sectors.

```
recover oldbook.txt
recover B:
```

rem [*remark*]

(Internal) Includes the specified *remark* in a batch file.

```
rem This file is called Billfile.bat.
```

ren *pathname filename*

(Internal) Changes the name of the file specified by *pathname* to *filename*.

```
ren b:\sales\region1  region2
```

replace *source pathname* [*target pathname*][**/a**][**/p**][**/r**] [**/s**][**/w**]

(External) Updates previous versions of files.

source pathname	the drive or directory that contains the replacement files. It can also be a single file or a wildcard filename.
target pathname	the drive or directory that contains the files you want to replace.
/a	adds files that exist in the source directory, but not in the target directory, to the target directory. Do not use **/a** with **/s**.
/p	prompts before replacing a target file or adding a source file.
/r	replaces read-only files as well as unprotected files.
/s	searches all subdirectories of the target directory while replacing matching files. Do not use **/s** with **/a**.
/w	causes **replace** to wait for you to press any key before it replaces files.

```
replace a:\phones.cli c:\ /s
```

restore *source drive:target pathname*[*pathname*][**/s**][**/p**][**/b:***date*] [**/a:***date*][**/e:***time*][**/l:***time*][**/m**] [**/n**]

(External) Restores one or more files previously backed up using the Backup command. You can restore files from one type of disk to another, such as from a diskette to a hard disk or from one type of diskette to another.

source drive:	specifies the drive that contains the backed up files.
target pathname	specifies the directory to which you want to restore the files.
pathname	specifies the disk directories and/or file you want to restore.
/s	restores the specified directory and its subdirectories.
/p	prompts for permission before restoring hidden or read-only files and before restoring any files changed since the last backup.
/b:*date*	restores only those files last modified on or before date *(mm/dd/yy)*.
/a:*date*	restores only those files last modified on or after date *(mm/dd/yy)*.
/e:*time*	restores only those files last modified at or before time *(hh:mm)*.
/l:*time*	restores only those files last modified at or after time *(hh:mm)*.
/m	restores only those files modified since the last backup.
/n	restores only those files that no longer exist on the target drive.

```
restore a: c:\*.dat /n
```

rmdir *pathname*
rd *pathname*

(Internal) Removes from the disk the subdirectory specified by *pathname*.

```
rmdir \bin\user\jim
```

select [[*drive1*:][*drive2*:][*path*]][*yyy*][*xxx*]

(External) Installs MS-DOS on a new diskette with the specified country information and keyboard layout. **Select** creates the necessary Config.sys and Autoexec.bat files on the new disk. (Requires international drivers.)

drive1:	the source drive.
drive2:	the target drive.
path	the directory name on the target drive in which to copy system files.
yyy	specifies the country code.
xxx	specifies the keyboard code.

```
select b: a: 049 gr
```

set [[*string1*]=[*string2*]]

(Internal) Sets *string1* equal to *string2* in the environment for use in later programs and batch files. Displays the **set** values if you omit all parameters. Including the *string1* parameter without the *string2* parameter removes the *string1* name from the environment.

string1	any character string you want to replace.
string2	any replacement character string.

```
set drive=b:
set pathname=c:\sales
```

shift

(Internal) Lets you use more than the usual ten replaceable parameters (%0-%9). Each parameter definition shifts up one place.

```
shift
```

shiptrak

(External) Parks the heads of a hard disk at the innermost tracks in preparation for moving the drive unit.

```
shiptrak
```

spooler [*/printer*][*/p*][*/s*][*/c*][*/g*]

(External) Lets you send commands to and get the status of the print spooler, assuming the Spooler.sys device drive has been loaded.

/printer	is the number of the printer you want to use. It must be 1 for the 1400 FD/HD.
/p	pauses the spooler.
/s	interrupts the spooler's buffering so that your computer immediately stops printing data in the buffer.
/c	clears the spooler.
/g	returns the status of the spooler (installed, pause on/off, buffering on/off, size of buffer, percentage of buffer full).

```
spooler /g
```

sort [**/r**][/+*n*][<<*input pathname*][>*output pathname*]

(External) Reads input from the keyboard or the file specified by *input pathname*, sorts the data, and writes it to the screen or to the file specified by *output pathname*.

/r	reverses the sort (sorts from Z to A).
/+*n*	begins the sort at column *n*. The default is column 1.

```
sort /r <unsort.txt >sort.txt
```

subst [*drive*:][*pathname*][**/d**]

(External) Substitutes a virtual *drive* name for a *pathname*.

drive:	the virtual drive name. The highest available drive letter is the one specified in Config.sys with the **lastdrive** command. The default is Drive E.
pathname	the pathname you want to replace.
/d	deletes the association between *drive*: and *pathname*.

```
subst d: b:\sales\region1
```

sys *drive*:

(External) Transfers the MS-DOS system files from the current disk to the disk in the specified *drive*.

```
sys b:
```

time [*hh:mm*[**:***ss*[*.cc*]]]

(Internal) Displays or sets the time.

hh:mm:ss.cc	specifies the time in hours, minutes, seconds, and hundredths of a second. Omit the time to display the time.

```
time 14:30
```

tree [*drive*:][**/f**]

(External) Displays all directories and subdirectories on the specified *drive*.

/f causes **tree** to also display all files on the drive.

```
tree b: /f
```

type *pathname*

(Internal) Displays the contents of the file specified by *pathname*.

```
type b:testfile
```

ver

(Internal) Displays the version number of your MS-DOS operating system.

```
ver
```

verify [on][off]

(Internal) Enables or disables disk write verify. Displays the current **verify** setting if you omit the parameters.

```
verify on
```

vol [*drive*:]

(Internal) Displays the volume label of the disk in the current or specified drive.

```
vol b:
vol
```

xcopy *source pathname* [*target pathname*][**/a**][**/d:***mm/dd/yy*][**/e**] [**/m**][**/p**][**/s**][**/v**][**/w**]

(External) Copies files and directories, including subdirectories. You can use **xcopy** to back up between different drive or media types.

source pathname	specifies the drive, directories, and/or files to copy.
target pathname	specifies the drive, directories, and/or file to copy to. If you omit this parameter, **xcopy** copies to the current directory. The default filename is *.*.
/a	copies only those files that have the archive bit set, without modifying the archive bit.
/d:*mm/dd/yy*	copies files modified on or after the specified date.
/e	copies any subdirectories, even if they are empty.
/m	copies only those files that have the archive bit set, and modifies the source files by turning off the archive bit.
/p	prompts you with `Y/N?` before copying each source file.
/s	copies directories and subdirectories, unless they are empty. If you omit **/s**, **xcopy** works within a single directory.
/v	verifies each target file as it is written to be sure it is identical to the source file.
/w	waits before copying the files. At the message, press any key to continue, or press **CTRL-C** to cancel **xcopy**.

```
xcopy a: b: /s /e
```

Special Keys

→	**Display Template**. Displays the contents of the keyboard template (a storage place for the last line you typed) one character at a time.
CTRL	**Execute Special Commands.** Lets you press only two or three keys to generate complex commands. Hold **CTRL** while pressing another key.
CTRL-C	**Terminate Process.** Stops the execution of a program if the program uses MS-DOS functions. Otherwise, this key sequence is not recognized. The computer might take a few moments before it recognizes the key sequence.
CTRL-H	**Correct Typing Error.** Moves the cursor left one character. Erases the character beneath the cursor.
CTRL-J	**End Line.** Causes the current line to end and performs a carriage return but does not cause the line to process. You can continue typing the command line.
PAUSE OR CTRL-S	**Stop Scroll.** Halts scrolling and lets you view the display. Press the space bar to continue scrolling.
CTRL-P	**Print Output.** Causes all computer output to print on your printer if it is connected and ready. Press this sequence again to stop the function.
CTRL-ALT-DEL	**Reset** (reboot) your computer.
CTRL-ALT-INS	**Setup mode.** Displays the setup menu.
CTRL-ALT-F11	Enable external video monitor.
CTRL-ALT-F12	Enable internal video monitor.

DEL	**Erase Character.** Removes the character under the cursor. If the cursor is at the first character of the line, this key does nothing.
ENTER	**Process Command: Generate Carriage Return.** Starts the processing a command line. **ENTER** also causes a carriage return; the cursor drops one line and returns to the left margin.
ESC	**Terminate Line.** Ends the current line but does not process it. **ESC** clears the line buffer, outputs a backslash, carriage return, and line feed but does not affect the keyboard template. The system prompt is not displayed, but the system is ready for a command.
F1	**Display Character.** Displays the next character in the keyboard template until you reach the end of the line.
F2 *char*	**Copy to Character.** Copies all characters up to the specified character (*char*) and displays them.
F3	**Display Template.** Redisplays the last typed command line. Execute the line again by pressing **ENTER** or edit the line before re-executing it.
F4 *char*	**Delete to Character.** Deletes all characters up to the specified character (*char*) from the keyboard template. They are skipped and not copied to the command line.
F5	**Replace Template.** Makes the line you type the new template but does not execute the command.
F6 or **CTRL-Z**	**End-of-File**. Puts an end-of-file character in the keyboard template.

INS	**Insert Text.** Lets you insert characters in a command line. To begin the insertion, press **INS**. Press **INS** again to end the insertion. Use the arrow keys to position the cursor before you begin inserting.
space bar	**Add a Space.** Moves the cursor one space to the right, and adds a space to a line.
PRINTSCRN	**Print Screen**. Causes the current screen display to print if your printer is connected and ready.

Edlin Commands

append lines

[*number*]**a**

Adds the specified *number* of lines from disk to memory. If you omit *number*, Edlin appends lines until available memory is 75% full.

```
100a
```

copy lines

[*line1*][*line2*][*line3*][*count*]**c**

Copies all lines in the range *line1* to *line2*, and places them immediately ahead of *line3* for the number of times specified by *count*.

```
3,9,12c
,20,35c
```

delete lines

[*line1*][,*line2*]**d**

Deletes all lines in the range *line1* to *line2*. Deletes the current line if you omit *line1* and *line2*.

```
5,25d
4d
,4d
```

edit line

line

Displays the specified *line* for editing.

```
4
```

end edit

e ENTER

Ends the Edlin program and saves the edited file.

```
e
```

insert

[*line*]**i**

Inserts lines of text immediately before the specified *line*, or enters lines into a new file. Omit *line* or include a period to use the current line. Include a number sign (#) to append the lines to the end of the file.

```
3i
 i
#i
```

list

[*line1*][*line2*]**l**

Displays all lines in the range *line1* to *line2*.

```
2,5l
26l
,10l
```

move lines

[*line1*][*line2*][*line3*]**m**

Moves all lines in the range *line1* to *line2* to the line immediately preceding *line3*.

```
23,30,100m
```

page

[*line1*][*line2*]**p**

Pages through a file 23 lines at a time, or lists the specified block of lines.

```
10,15p
20p
```

quit

q

Quits the editing session without saving the file.

```
q
```

replace string

[*line1*][,*line2*][**?**]**r**[*string1*] **CTRL-Z** [*string2*]

Replaces all occurrences of *string1* with *string2* in the lines between *line1* and *line2*. The question mark (?) causes Edlin to let you verify each modification.

```
2,7?rHi CTRL-Z Bye
```

search text

[*line1*][,*line2*][**?**]**s**[*string*]

Searches all lines in the range *line1* to *line2* for each occurrence of the text *string*. The question mark (?) prompt appears at each occurrence of *string*.

```
1,10sand
```

transfer lines

[*line*]**t**[*drive*:]*filename*

Inserts the contents of the file specified by *filename* immediately ahead of the specified *line* or the current line in the file being edited.

```
10tb:myfile
```

write lines

[*number*]**w**

Writes a specified *number* of edited lines from memory to disk, beginning with Line 1. If you omit *number*, Edlin writes until 25% of memory is freed.

```
100w
```

Edlin Editing Keys

Function	Key(s)	Description
Copy *char*	→	Copies one character to the new line.
Copy to *char*	**F2** *char*	Copies all characters up to the specified character to the new line.
Copy all	**F3**	Copies all remaining characters in the template to the new line.
Delete *char*	**DEL**	Deletes a character in the template.
Delete to *char*	**F4** *char*	Deletes a number of characters up to, but not including, the specified character (*char*).
Void line	**ESC**	Voids or terminates the current line entry.
Insert	**INS**	Enters or exits the insert mode.
Replace template	**F5**	Replaces the template with the characters displayed to allow further editing.
Void line	**F6**	Cancels any changes made to the current line. The template is unchanged and you can continue making changes.
Enter line	**ENTER**	Makes the new line the new template and sends it to the requesting program.

You can use the following symbols in Edlin:

. (period)	Indicates the current line.
#	Indicates the end of a program.
+	References lines after the current line.
-	References lines before the current line.

Debug Command Parameters

The following parameters are available for the debug commands listed in the next section.

address — An alphabetic segment register and an offset, such as:

`cs:0100` — A segment address and offset.

`04ba:0100` — An offset only. (The default segment is cs for the **go**, **load**, **trace**, **unassemble**, and **write** commands, and ds for all other commands.)

byte — A 2-digit hexadecimal value placed in or read from an address or a register.

drive — A 1-digit value for the drive to be used for accessing or writing data, as follows:

0 = Drive A 2 = Drive C
1 = Drive B 3 = Drive D

pathname — A drive specification, filename, and extension. (The complete pathname is optional; however, you must specify at least the filename.)

list — A series of strings or byte values. (*List* must be the last parameter.)

range — An area of memory specified by:

address1 address2 (*Address2* must be an offset.)

address1 value (*Value* is the number of lines on which the command operates. The default is 80.)

Do not use the *address1value* format if another hexadecimal value follows the *range* parameter.

registername — One of the following registers:

ax	cs	bp	di
bx	ds	ip	si
cx	es	sp	pc
dx	ss	f	

sector — A hexadecimal value (1-3 characters), indicating the relative sector number on the disk.

sectorcount — A hexadecimal value (1-3 characters), indicating the number of disk sectors to write or load.

string — Any number of characters, enclosed in quotation marks (single or double).

value — A hexadecimal value (4 characters maximum).

Debug Commands

assemble

a [*address*]

Assembles statements directly into memory, starting at *address*.

```
a  cs:0100
```

compare

c *range address*

Compares the portion of memory specified by *range* to a portion of the same size beginning at the specified *address*, and displays all differences.

```
c 100,1ff 300
c 100 l 100 300
```

dump

d [*address*]
d [*range*]

Displays the contents of the specified memory *address* or *range*.

```
d cs:100 109
```

enter

e *address* [*list*]

Enters byte values into memory at the specified *address*; replaces the contents of memory, beginning at *address*, with the *list* of values.

```
e ds:100 45 a1 "abc" 0f
e cs:1004
```

fill

f *range list*

Fills the memory locations in the specified *range* with the values in the *list*.

```
f 04ba:100 l 100 42 45 52 54 41
```

go

g [=*address1*[*address2*...]]

Executes the program currently in memory, beginning at *address1* and stopping at each breakpoint (specified by *address2*...).

```
g=cs:7550
```

hex

h *value1 value2*

Displays the results of *value1*+*value2* and *value1*-*value2* (hexadecimal arithmetic).

```
h  19f  10a
```

input

i *value*

Inputs and displays one item from the port specified by *value*.

```
i 2f8
```

load

l [*address*[*drive*: *sector sectorcount*]]

Loads the specified file from the *drive* (0-3) into memory, beginning at the specified *address*. Loads absolute sectors from the *drive*, beginning at *sector* and continuing until the number of sectors specified by *sectorcount* is loaded.

```
l 04ba:100 2 0f 6d
```

move

m *range address*

Moves the block of memory specified by *range* to the location beginning at *address*.

```
m cs:100 110 cs:500
```

name

n *pathname1* [*pathname2...*]

Assigns program names for later load or write commands and assigns pathname parameters for the file being debugged.

```
n file1.exe
n file2.dat file3.dat
```

output

o *value byte*

Sends the specified *byte* to the port specified by *value*.

```
o 2fb 4f
```

proceed

p [=*address*][*value*]

Beginning at *address*, proceed executes the number of instructions specified by *value*. The purpose of this is to execute all the instructions associated with call, int, or loop – or to repeat string instructions – and then stop execution at the next instruction. After it executes each instruction, proceed displays the register contents, flags, and next instruction. If you omit all parameters, proceed causes the execution of the instruction pointed to by CS:IP.

```
p
p=011a 10
```

quit

q

Quits debug without saving the file.

```
q
```

register

r [*registername*]

Displays the contents of all registers and flags, or displays the one register or the flags specified by *registername*, and lets you change the setting.

```
r
rax
rf
```

search

s *range list*

Searches the location in the specified *range* for the *list* of bytes.

```
s  cs:100 110 41
```

trace

t [=*address*][*value*]

Executes one or more instructions (specified by *value*), beginning at *address*. The trace command displays the register contents, flags, and the next instruction after each instruction executes.

```
t
t=011a 10
```

unassemble

u [*address*]
u [*range*]

Disassembles instructions beginning at the specified *address* (or for the specified *range*), and displays their addresses, their hexadecimal values, and the source statements that correspond to them.

```
u 04ba:100 l 10
```

write

w [*address*[*drive*: *sector sectorcount*]]

Writes the data being debugged to a disk file on the specified *drive* (0-3), beginning at the specified *address*. Writes absolute sectors to the specified *drive*, beginning at *sector*, and continuing until the number of sectors specified by *sectorcount* is written.

```
w cs:100 1 37 2b
```

GW-BASIC
Quick Reference

Loading GW-BASIC

Use the following syntax to load GW-BASIC at the MS-DOS system prompt:

basic | **basica** [*pathname*] [<*input file*][>[>]*output-file*] [**/f:***max. # of files open at same time*][**/c:***RS-232 receive buffer size*] [**/m:***highest memory location for BASIC, max. block size*][**/s:***max. record length for direct access files*][**/d**][**/i**]

pathname	loads and executes the specified GW-BASIC program file.
<*input file*	inputs data from the specified file instead of from the keyboard.
output file	outputs data to the specified file instead of to the video display. Use > to overwrite the existing output file or >> to append to it.
/f:*files*	specifies the maximum number of data files GW-BASIC can open at one time, including the files it reserves for internal use. The maximum value you can specify for files is 15. The default is 3. Use a **files** command in your Config.sys file if you want a number other than the default. You **must** use the **/i** option when using the **/f** parameter.
/c:*buffer size*	sets the size of the RS232 receive buffer. The default is 256 bytes. (The RS232 transmit buffer is always set to 128 bytes.)
/m:*location, block size*	loads GW-BASIC with the reserved memory specified by *block size* (block size x 16). GW-BASIC uses memory up to the memory *location*, and memory above is reserved for machine language routines. The default is 64K bytes for GW-BASIC (4096 blocks of 16 bytes each).

/s: *record length*	sets the maximum length of any record in a file created for direct access. The default is 128 bytes.
/d	loads the double-precision transcendental math package.
/i	causes GW-BASIC not to dynamically allocate space during file operations. This switch is always invoked if you use BASICA.

GW-BASIC Commands and Statements

This chapter contains an alphabetical reference to all GW-BASIC statements and functions. At the beginning of each statement or function description is a syntax line that gives the format for typing the statement or function. Each syntax line observes the following notations:

bold	specifies a command, statement, or parameter that you must type exactly as it appears.
lowercase italics	represent variable names, letters, characters, or values.
[ ]	indicates that you can use optional parameters.
... (ellipsis)	indicates that you can repeat a parameter.
| (vertical bar)	indicates an either/or situation.
SMALL CAPS BOLD	indicates a key combination, such as **CTRL-C**. Press the second key while holding down the first key.

abs(*number*)

Computes the absolute value of *number*.

```
print abs(-44)
x=abs(y)
```

asc(*string*)

Returns the ASCII code (a decimal number) for the first character of *string*.

```
print asc("a")
n=asc(b$)
```

atn(*number*)

Computes the arctangent of *number* in radians.

```
print atn(7)     x=atn(y/3)*57.29578
```

auto [*line*][,*increment*]

Automatically generates a line number when you press **ENTER**. If *line* exists in memory, GW-BASIC displays an asterisk after the number. To turn off **auto,** press **BREAK**.

line	is the starting line number. Default = Line 10.
increment	is the increment to use when generating line numbers.

```
auto
auto 100,50
```

beep

Produces a sound at 800 Hz for 1/4 second.

```
beep
```

bload *pathname* [,*offset*]

Loads a memory image file, specified by *pathname,* into memory.

offset	is the location (the number of bytes from the beginning of the current segment) where GW-BASIC loads the image.*offset* must be in the range 0-65535. Default = the value set by **bsave**.

```
bload "prog1.txt"
bload "prog2.txt",0
```

bsave *pathname,offset,length*

Saves the contents of an area of memory into a disk file (*pathname*).

offset — is the location (the number of bytes from the beginning of the current segment) where GW-BASIC starts saving. It must be in the range 0-65535.

length — is the number of bytes to save (in the range 1-65535).

```
bsave "prog1.sav",0,500
```

call *variable* [(*parameter list*)]

Transfers program control to an assembly-language subroutine stored at *variable*.

parameter list — contains the variables that are passed to the external subroutine.

```
call c
call c (a$,z,x)
```

cdbl(*number*)

Converts *number* to double precision.

```
print cdbl(465.342)
z=cdbl(a)
```

chain [**merge**]*pathname*[,[*line*][,**all**][,**delete** *line*-*line*]]

Lets the current program load and execute another program (specified by *pathname*). You must have first saved *pathname* in ASCII format. Commas in the syntax line are significant and must be entered even if you omit the option.

line — is the line number at which execution begins in the chained program. Default = the first line of the chained program.

all passes every variable in the current program to the chained program. If you omit all, the current program must contain a **common** statement to pass variables to the chained program.

merge overlays the lines of the chained program with the current program.

delete deletes lines in the overlay that you can merge in a new overlay.

```
chain "prog2"
chain "subprog.bas",,all
```

chdir *pathname*

Changes the directory to the directory specified by *pathname*.

```
chdir "b:\accts\recvble"
chdir ".."
```

chr$(*code*)

Returns the character corresponding to any ASCII or control code.

```
print chr$(35)
c$=chr$(32)
```

cint(*number*)

Converts *number* to an integer representation by rounding the fraction portion of the number.

number must be in the range -32768 to 32767.

```
print cint(1.6)
z=cint(-1.67)
```

circle [step] (*x, y*),*radius*[,*color*[,*start*,[*end*[,*aspect*]]]]

Graphics. Draws an ellipse, the center of which is (*x, y*), on the screen.

step	designates (*x, y*) as relative coordinates.
radius	is the major axis of the ellipse.
color	designates the color of the ellipse. *Color* must be a valid number in the current color set.
start and *end*	are the beginning and ending angles, in radians (in the range -6.283186 to 6.283186).
aspect	is the ratio of the x-radius coordinate to the y-radius coordinate.

```
circle (150,100),50
```

Note: Color functions only when using an external color monitor. With the built-in LCD, this parameter will affect only screen gradation.

clear [,[*memory location*] [,[*stack space*][,*video memory*]]]

Frees memory for data without erasing the program in memory.

memory location	specifies the highest memory location available for GW-BASIC.
stack space	specifies the memory to set aside for temporarily storing internal data and addresses.
video memory	specifies the amount of memory to be set aside as video memory.

```
clear
clear ,45000
clear ,6100,300
clear ,,,32768
```

close [*buffer*,...]

Closes access to a disk file or communications channel.

buffer buffer number of file to close. Default = Close all files.

```
close
close 1,2,8
```

cls

Clears the screen (or active viewport) and returns the cursor to the home position (the upper left corner of the screen or viewport).

```
cls
```

color [*background*] [,*palette*]

Graphics. Selects the background color and the palette for Screen Mode 1.

background specifies the background color.

palette specifies the palette to use for the foreground colors.

```
color 0,7
color 0,1
```

Note: *Color* functions only when using an external color monitor. With the built-in LCD, this parameter will affect only screen gradation.

color [*foreground*][,*background*][,*border*]]

Text Mode Only. Selects the display colors for the foreground, background, and border for Screen Mode 0.

foreground specifies the color of the foreground characters. Foreground can be colors 0 to 15 of the color set. Specify color +15 to get blinking characters.

background specifies the background color and can be colors 0 to 15 of the 16 color set.

border specifies the border color. Border can be 0 to 15.

```
color 0,7
color 1,0
```

com(*channel*) *action*

Turns on, turns off, or temporarily halts the trapping of communications channel activity.

channel can be 1 or 2.

action can be on, off, or stop.

```
com(1) on
com(2) stop
```

common *variable* [,*variable*...]

Reserves space for variables so you can pass them to a chained program. Be sure both programs in the chain contain **common** statements.

```
common a, b$, d()
```

cont

Resumes execution when either **CTRL-BREAK** or the execution of a **stop** or an **end** statement halts a program.

```
cont
```

cos(*number*)

Computes the cosine of *number*.

```
print cos(5.8)
y=cos(x*.00174533)
```

csng(*number*)

Converts *number* to single precision. GW-BASIC rounds the number when converting it to single-precision.

```
print csng(.145388509)
z=csng(a#)
```

csrlin

Returns the current row position of the cursor.

```
print csrlin
a=csrlin
```

cvd(*8-byte string*)
cvi(*2-byte string*)
cvs(*4-byte string*)

Converts a string to a double-precision (**cvd**), an integer (**cvi**), or a single-precision (**cvs**) number. Use these functions to restore data to numeric form after reading from the disk.

```
a# = cvd(grosspay$)
d = cvs(total$)
```

data *constant* [,*constant...*]

Stores numeric and string constants to be accessed by a READ statement. Enclose string constants containing delimiters (leading or trailing blanks, colons, commas) in quotation marks.

```
data Edmonton, New York, Chicago, Brooks
```

date$[=*string*]

Sets the date to the specified string, or retrieves the current date if you omit *string*.

```
date$="04/17/85"
today$=date$
```

defdbl *letter*[,*letter*...]
defint *letter*[,*letter*...]
defsng *letter*[,*letter*...]
defstr *letter*[,*letter*...]

Defines variables as double precision (**dbl**), integer (**int**), single precision (**sng**), or string (**str**).

```
defdbl a
defint a-g
defstr h-m
```

def fn*name* [(*argument list*)] = *expression*

Defines *name* as a function according to the expression.

name is a valid variable name.

argument list is a list of dummy variables used in *expression*.

expression defines the operation to perform.

```
def fnr = rnd(1)*89+10
```

def seg[=*address*]

Assigns the specified address to be the current segment address.

address is in the range 0-65535. Default = GW-BASIC's Data Segment.

```
def seg
def seg = &HB800
```

def usr[*number*]=*offset*

Defines the user number and segment offset of a subroutine to be called by the **usr** function.

number is in the range 0-9. Default = 0.

offset is in the range 0-65535.

```
def usr = 0
def usr3 = &h0020
```

delete *line1*[-*line2*]

Deletes *line1* through *line2* of the program in memory.

line1	defaults to the first line of the program.
line2	defaults to the last line of the program.

```
delete 70
delete -110
```

dim *array* (*dimension*)[,*array*(*dimension*)...]

Sets aside storage for arrays with the dimensions you specify.

```
dim ar(100)
dim l1%(8,25)
```

draw *string*

Graphics. Draws an image on the screen.

string	specifies one or more of the movement commands. Prefix commands can precede the movement commands.

Movement Commands

u[*n*]	Moves up *n* points.
d[*n*]	Moves down *n* points.
l[*n*]	Moves left *n* points.
r[*n*]	Moves right *n* points.
e[*n*]	Moves diagonally up and right *n* points.
f[*n*]	Moves diagonally down and right *n* points.
g[*n*]	Moves diagonally down and left *n* points.
h[*n*]	Moves diagonally up and left *n* points.
m*x,y*	Moves to point *x,y*. If you precede *x* with a plus (+) or minus (–) sign, **draw** assumes it is a relative position. Otherwise, it is an absolute position.

Prefix Commands

b	Plots no points after the move.
n	Returns to the original position when the move is complete.
a*angle*	Sets the angle of the move.
c*color*	Sets the color.
p*color,border*	Paints, using *color* and *border*.
s*factor*	Sets the scale factor.
t*angle*	Moves at the specified angle.
x*variable*	Executes a substring.

```
draw"u30;"+"d30;"+"140;"+"r40;"
```

edit *line*

Enter the Edit mode. GW-BASIC displays *line* for editing.

```
edit 100
edit .
```

end

Ends program execution, and closes all files.

```
end
```

environ "*parameter id* = *text*"[;"*parameter id* = *text*"...]

Lets you modify GW-BASIC's Environment String Table to change the **path** parameter for a child process or to pass parameters to a child process.

parameter id	is the name of the parameter. You must enclose it in quotation marks and type it in uppercase characters.
text	is the new parameter text. You must use an equal (=) sign to separate it from *parameter id*. If you omit *text*, or specify a null string or a semicolon, GW-BASIC removes the parameter from the Environment String Table and compresses the table.

```
environ "path=a:\"
```

```
environ "sales=mysales"
environ "path=a:\";"sales=mysales"
```

environ$ [("*parameter id*")][(*number*)]

Returns the specified environment string from GW-BASIC's Environment String Table. *Number* and *parameter id* are mutually exclusive. You can specify only one on the command line.

parameter id — is the parameter for which to search. It must be uppercase and enclosed in quotation marks.

number — specifies which parameter to return by its position within the table.

```
print environ$("path")
print environ$ (3)
```

eof(*buffer*)

Function. Detects the end of a file.

buffer — is the number assigned to the file when you opened it. In sequential access files, **eof** returns 0 (false) if end-of-file has not been read or -1 (true) if it has. In direct access files, **eof** returns -1 (true) if the last executed **get** statement was unable to read an entire record because of an attempt to read beyond the physical end of the file.

```
if eof(1) then goto 1540
```

eof(*buffer*)

Communications. Detects an empty input queue for a communications file.

buffer — is the number assigned to the file when you opened it. In ASCII mode, **eof** returns -1 (true) if **CTRL-Z** is received. In binary mode, **eof** returns -1 (true) when the input queue is empty.

```
if eof(1) then return
```

erase *array* [,*array*...]

Erases one or more arrays from memory. Lets you either redimension arrays or use their previously allocated space in memory for other purposes.

```
erase c
erase g,h,i,z$
```

erdev

Returns the value of a device error within MS-DOS as set by the Interrupt 24 handler. The lower 8 bits of **erdev** contain the Interrupt 24 error code.

```
erdev
```

erdev$

Returns the name of the device (as set by the Interrupt 24 handler) when a device error occurs. A character device error returns a 8-byte device name. A block device error returns a 2-byte device name.

```
erdev$
```

erl

Returns the number of the line in which an error has occurred. If no error has occurred, **erl** returns 0. If the error occurs while you are entering something at the prompt, **erl** returns 65535 (the largest number than can be represented in two bytes).

```
print erl
e=erl
```

err

Returns the error code if an error has occurred.

```
if err = 7 then 1000 else 2000
```

error *code*

Simulates a specified error.

code is one of GW-BASIC's error codes.

```
error 1
```

exp(*number*)

Returns the natural exponent of *number*, that is e (base of natural logarithms) to the power of *number*.

number must be less than or equal to 88.02968.

```
print exp (-2)
a=exp(-6)
```

exterr (*number*)

Returns extended error information.

number is an integer in the range 0-3 as follows:

0 = Extended error code
1 = Extended error class
2 = Extended error suggested action
3 = Extended error location

```
exterr(0)
print exterr(3)
```

field *buffer*, *length* **as** *variable* [,*length* **as** *variable*...]

Divides a direct access buffer into fields so you can send data from memory to disk and from disk to memory. You identify each field by a string variable and you specify its length.

length must be an integer in the range 1-255.

```
field 3, 128 as a$, 128 as b$
```

files[*pathname*]

Displays the names of the files and directories on a disk.

pathname lists all files that match *pathname*. If you omit the filename from *pathname*, GW-BASIC lists all files and directories in the specified directory.

```
files
files"\books\"
```

fix(*number*)

Returns the truncated integer of *number*.

```
print fix(2.6)
z=fix(b)
```

for *variable* = *initial value* **to** *final value* [**step** *increment*] **next** [*variable*]

Establishes a program loop that allows a series of program statements to execute a specified number of times.

increment is the number added to *initial value* each time the loop executes. Default=1

variable must be either integer or single-precision.

```
for x =1 to 5 : print x : next
```

fre(*dummy argument*)

Returns the number of bytes in memory not being used by GW-BASIC. Specify a string argument if you want GW-BASIC to compress the data before returning the amount of memory available.

```
print fre(44)
print fre("44")
```

get[#]*buffer*[,*record*]

Reads a record from a direct access disk file, and places it in the specified *buffer*.

record is an integer in the range 0-16,777,215. If you omit *record*, GW-BASIC gets the next sequential record.

```
get 1
get 1,25
```

get[#]*buffer*,*number*

Communications. Transfers data from the communications line to the communications *buffer*.

number specifies the number of bytes to transfer.

```
get 1,8
```

get(*x1*, *y1*)-(*x2*, *y2*),*array*

Graphics. Transfers points from an area on the display to an array.

(*x1*,*y1*) and (*x2*,*y2*) are the coordinates at which the image begins and ends.

array is a numeric array to hold the image.

```
get (0,0) - (100,100),z
```

gosub *line*

Branches to the subroutine, beginning at *line*. Every subroutine must end with a **return** statement.

line is the starting line number in your program of the subroutines.

```
gosub 1000
```

goto *line*

Branches to the specified line.

line is the line number in your program to branch to.

```
goto 100
if r = 14 then goto 80
```

hex$(*number*)

Calculates the hexadecimal value of *number.*

number specifies the decimal value of the range -32768 to +65535 to convert into a hexadecimal string.

```
print hex$(30)
y$=hex$(x/16)
```

if *expression(s)* **then** *statement(s)* **[else** *statement(s)***]**

Tests a conditional *expression*, and makes a decision regarding program flow. If *expression* is true, GW-BASIC executes the **then** statement. If *expression* is false, GW-BASIC executes the matching **else** statement or the next program line.

expression is any numeric or string expression, usually making logical or relational comparisons.

statement(s) can be one or more valid BASIC statements.

```
if a=b then print "a=b" else print "a<>b"
```

inkey$

Returns a 0-, 1-, or 2-byte string from the keyboard without requiring that you press **ENTER**. **Inkey$** does not echo the character you press.

0-byte string = no key pressed
1-byte string = the ASCII value of the key
2-byte string = 00 in Byte 1, extended ASCII value of the key in Byte 2.

```
10 a$ = inkey$:if a$ = "" then 10
```

inp(*port*)

Returns the byte read from *port*.

port can be any integer in the range 0-65535.

```
a = inp(255)
```

input[;]["*prompt*";]*variable*[,*variable*...]

Inputs data from the keyboard into one or more variables. GW-BASIC stops execution, displays the message specified by *prompt*, followed by a question mark, and then waits for input. If you do not want GW-BASIC to display the prompt, type a comma instead of a semicolon after *prompt*.

; immediately following **input**, echoes **ENTER** when you press it as part of a response.

```
input y%
input "Enter your name and age";n$, a
```

input # *buffer*, *variable*[,*variable*...]

Inputs data from a sequential device or file, and stores it in a program variable.

buffer is the number BASIC assigned to the file when opening it.

```
input #1,a,b
input #4,a$,b$,c$
```

input$(*number*[,[#]*buffer*])

Inputs a string of characters from either the keyboard or a sequential access file.

number specifies the number of characters to input. It can be in the range 1-255.

buffer tells GW-BASIC to input the string from a sequential access file. If you omit *buffer*, GW-BASIC inputs the string from the keyboard.

```
a$ = input$(5)
a$ = input$(11,3)
```

instr([*number*,]*string1*,*string2*)

Searches for the first occurrence of *string2* in *string1*, and returns the position at which the match is found.

number specifies the position in *string1* at which to begin searching. Default = the first character in *string1*.

```
instr (3, "1232123", "12")
a$ = "Lincoln":instr(a$,"inc")
```

int(*number*)

Converts *number* to the largest integer that is less than or equal to *number*.

number is not limited to the integer range.

```
print int(79.89)
print int (-12.11)
```

ioctl [#]*buffer*,*string*

Sends a control data string to a device driver.

string is a string expression containing a series of commands. The commands are separated by semicolons. *string* can be a maximum of 255 bytes.

```
ioctl #1,"pl56"
```

ioctl$([#]*buffer*)

Returns the control data string from a device driver that you have previously opened.

buffer is the number assigned to the driver when you opened it. The number sign (#) is optional.

```
if ioctl$(1) = "nr" then print "Printer
not ready"
```

key *number,string*

Assigns function key values.

number — specifies a function key (**F1-F10**) or a user key (15-20). See **key**(*number*) *action*.

string — is the string expression assigned to the key. It can contain a maximum of 15 characters.

```
key 1,"John Seymore #3346"
```

key on | off | list

Displays the function key assignment values.

on — displays the key assignments on Line 25 of the screen.

off — erases the key assignments.

list — displays all 15 characters of the soft key assignments

key(*number*) *action*

Turns on, turns off, or temporarily halts key trapping for a specified key. GW-BASIC numbers the cursor direction keys 11-14, and the user-defined keys 15-20.

action — can be on, off, or stop.

number — can be a number in the range 1-20, indicating the number of the key to trap. Function keys use their corresponding function key number (1-10).

Use the following syntax to define your own user keys:

```
key number,CHR$(key) + CHR$(scan)
```

kill *pathname*

Kills (deletes) *pathname* from disk.

```
kill "file.bas"
kill "a:\report\data"
```

left$(*string*,*number*)

Returns the specified number of characters from the left portion of *string*.

number is an integer in the range 1-255.

```
print left$("battleships",6)
```

len(*string*)

Returns the total number of characters (including blanks) in *string*.

```
x = len(sentence$)
print len("dog") + len("terrier")
```

[let]*variable=expression*

Assigns the value of expression to *variable*.

```
let a$ = "a rose is a rose"
b1 = 1.23
```

line [(*x1*,*y1*)]-[**step**](*x2*,*y2*),[*color*][,**b** [**f**]][,*style*]

Graphics. Draws a line or a box on the video display.

(*x1*, *y1*) is the point at which the line begins. Default = last specified screen point.

(*x2*, *y2*) is the point at which the line ends.

color specifies the color of the line.

b, f causes GW-BASIC to draw a box. The points that you specify are opposite corners. If you specify both the b and f options, GW-BASIC draws a box and fills the box in with *color*.

style is a 16-bit integer that lets you select the line-style used when drawing normal lines and unfilled boxes. If bit = 1, GW-BASIC draws the point. If bit = 0, it does not.

```
line (0,0)-(319,199)
line -(319,199)
```

Note: *Color* functions only when using an external color monitor. With the built-in LCD, this parameter will affect only screen gradation.

line input[;]["*prompt*";]*string variable*

Inputs an entire line (up to 254 characters) from the keyboard including delimiters (commas, quotation marks, etc). You terminate string input by pressing **ENTER**. However, if **line input** is immediately followed by a semicolon, pressing **ENTER** does not echo a carriage return to the display.

prompt is a string literal that is displayed on the screen.

string variable accepts all input characters from the keyboard.

```
line input a$
line input "Last name, first name?";n$
```

line input# *buffer*, *variable*

Inputs an entire line of data from a sequential file including delimiters (commas, quotation marks, etc).

buffer is the number assigned to the file when it was opened.

variable is the string variable name in which you want the data stored.

```
line input#1, a$
```

list [*startline*][-[*endline*]][,"*device*:"]

Lists a program in memory to the display or specified *device*.

startline specifies the first line to edit. Default = first program line.

endline specifies the last line to list. Default=last program line.

device: specifies the output device, scrn: (the screen) or lpt1: (the printer).

```
list
list 50-100,"lpt1:"
```

llist [*startline*][**-**[*endline*]]

Lists program lines in memory to the printer. **llist** assumes a132-character-wide printer. (You can change this using the **width** statement.)

startline specifies the first line to be listed.

endline specifies the last line to be listed.

```
llist
llist 68-90
```

load *pathname* [**,r**]

Loads a GW-BASIC program from disk into memory. The r option tells GW-BASIC to run the program.

pathname is the filename specification of the file you want to load.

```
load "a:prog1.bas"
load "prog1.bas",r
```

loc(*buffer*)

Returns the current record position within a file.
In direct access files, **loc** returns the record number accessed by the last **get** or **put** statement. In sequential access files, **loc** returns the number of 128-byte records that you have read or written.

```
a=loc(2)
if loc(1)55 then end
```

loc(*buffer*)

Communications. Returns the number of characters in the input queue. If more than 255 characters are in the input queue, **loc** returns 255. If there are fewer, **loc** returns the actual number of characters waiting to be read.

buffer is the number assigned to the file when you opened it.

```
if loc(x)>0 then 1000
```

locate [*row*][,[*column*][,[*cursor*][,[*start*][,*stop*]

Positions the cursor on the screen at *row* and *column*.

cursor sets the cursor. A value of 1 makes the cursor visible. A value of 0 makes the cursor invisible.

start is a numeric expression in the range 0-31 that specifies the first scan line of the cursor.

stop specifies the last scan line of the cursor, and it can be in the same range as *start*.

```
locate 10,20,1,
locate 25,1,1,3
```

lock [#]*buffer*[,*record*]
unlock [#]*buffer*[,*record*]

Controls access by other processes to all or part of an opened file, specified by *buffer*.

record is the record or the range of records to lock or unlock.

```
lock 1,1 to 4
unlock 1, 1 to 4
```

lof(*buffer*)

Returns the length of the file in bytes.

buffer is the number BASIC assigned to the file when opening it.

```
y = lof(5)
```

lof(*buffer*)

Communications. Returns the free space in the input queue. You can use **lof** to determine when an input queue is getting full.

buffer is the number assigned to the file when it was opened.

```
if lof(5)<20 goto 1000
```

log(*number*)

Computes the natural logarithm of *number*.

number must be greater than zero.

```
print log(3.14159)
z = 10 * log(p5/p1)
```

lpos(*number*)

Returns the logical position of the print head within the printer's buffer.

number can be 0 or 1 to indicate LPT1.

```
100 if lpos(x)>60 then lprint
```

lprint [**using** *format*;] *data*[,*data*...]

Prints data on the printer. **Lprint** assumes a print width of 132 characters You can change the width using **width.**

See **print** and **print using** for more information on formatting the output.

format is the one or more field specifiers enclosed in quotation marks.

data is a string or numeric value.

```
lprint (a * 2)/3
lprint using "#####.#";2.17
```

lset *field name = data*

Moves data to the direct access buffer, and places it in *field name* in preparation for a **put** statement. You must convert to a string any numeric value placed in a direct access file buffer with an **lset** statement. See **mks$, mkd$,** and **mki$.**

field name is a string variable defined in a **field** statement. **Lset** left-justifies the data. You must use **field** to set buffer fields before using **lset.**

```
lset ad$ = "2000 East Pecan St."
lset td$=e$
```

merge *pathname*

Loads a program and merges it with the program in memory. The file must be in ASCII format (saved with the **a** option).

pathname is the filename specification of the file you want to load.

```
merge "prog2.txt"
```

mid$(*oldstring*,*start*[,*length*])=*newstring*

Replaces a portion of *oldstring* with *newstring.*

start specifies the position of the first character you want to change.

length specifies the number of characters you want to replace.

```
mid$ (a$,3,4) = "12345":print a$
```

mid$(*string*,*start*[,*length*])

Returns a substring of *string*.

start specifies the position in the string from which to get the substring.

length is the number of characters in the substring. It must be in the range 1-255.

```
a$ = "Weatherford":print mid$(A$,3,2)
```

mkdir *pathname*

Creates the directory specified by *pathname*.

pathname is a standard directory specification.

```
mkdir "a:\accts\payable"
mkdir "\address"
```

mkd$(*double-precision expression*)
mki$(*integer expression*)
mks$(*single-precision expression*)

Converts numeric values to string values. **Mkd$** returns an 8-byte string; **mki$,** a 2-byte string; and **mks$,** a 4-byte string. These are the inverse functions of **cvd, cvi,** and **cvs.**

expression is an integer or single- or double-precision number.

```
lset ytd$ = mks$(564.33)
lset tot$ = mks$(tot)
```

name "*old filename*" as "*new filename*"

Renames *old filename* as *new filename*.

old filename is the name of the file to be renamed.

new filename is the new name of the file.

```
name "file.bas" as "file.old"
```

new

Deletes the program in memory, and clears all variables.

```
new
```

oct$(*number***)**

Returns a string that represents the octal value of a decimal number.

number specifies the decimal value to convert into an octal string.

```
print oct$(30) s$=oct$(90)
```

on com(*channel***)gosub** *line*

Transfers program control to a subroutine beginning at *line* when activity occurs on the specified communications channel.

channel specifies communications Channel 1 or 2.

line is the subroutine line at which execution begins when activity occurs on the communications channel. Specifying Line 0 turns off communications trapping.

```
on com (1) gosub 1000
```

on error goto *line*

Transfers control to *line* if an error occurs. You must execute an **on error goto** before the error occurs. Specifying Line 0 turns off error trapping.

```
on error goto 1500
```

on *number* **gosub** *list*

Causes a branch to the line specified by the value of *number*.

list is a list of one or more line numbers separated by commas. Therefore, if *number* equals 1, GW-BASIC branches to the first line in *list*. If *number* equals 2, GW-BASIC branches to the second line in *list*.

number must be in the range 0-255.

```
on y gosub 1000, 2000, 3000
```

on *number* **goto** *list*

Goes to the line specified by the value of *number*.

list is a list of one or more line numbers separated by commas. Therefore, if *number* equals 1, GW-BASIC branches to the first line in *list*. If *number* equals 2, GW-BASIC branches to the second line in *list*. *number* must be in the range 0-255.

```
on mi goto 150, 160, 170, 150, 180
```

on key(*number*)**gosub** *line*

Transfers program control to a subroutine beginning at *line* when you press the specified key.

number specifies the number of the key to trap. GW-BASIC numbers function keys 1-10, the cursor direction keys 11-14, and the user keys 15-20. You define the user keys with the **key** statement.

```
on key(13) gosub 500
```

on play(*number*)**gosub** *line*

Transfers program control to the subroutine beginning at *line* when the number of notes in the background music buffer goes from *number* to *number* minus 1.

number must be in the range 1-32.

```
on play(30) gosub 200
```

on timer(*number*)**gosub** *line*

Transfers program control to the subroutine at *line* when the specified time has elapsed.

number specifies the number of seconds. It can be a value in the range 1-86400 (86400 seconds = 24 hours).

```
on timer(3600) gosub 500
```

open *mode*,*buffer*,[*pathname*][*device*;] . . . . . [,*record length*]
open [*pathname*][*device*;][**for** *mode*][*access*] **as** *buffer*[**len**=*record length*]

Establishes an input/output path for a file or device.

In the first form of the syntax, you must use the abbreviated form of *mode* and must enclose it in quotation marks.

In the second form of the syntax, you must specify the complete word for *mode*. You cannot specify random. If you want to use direct access in the second form of the syntax, omit *mode*.

buffer specifies the I/O buffer (1-15) in memory to use when accessing the file. If you do not specify *pathname*, you must specify *device;*.

record length sets the record length for direct access files. It can be in the range 2-32767. Default = 128 bytes.

mode specifies any of the following:

o (open)	sequential output mode
i (input)	sequential input mode
a (append)	sequential extension of an existing file
r (random)	direct input/output mode

access controls the processes that can access the file and the degree to which they do so. *Access* can be shared, lock read, lock write, or lock read write.

```
open "r",2,"test.dat"
open "lpt1;"for output as 2
```

open "com *channel*: [*speed*] [,*parity*] [,*data*][,*stop*][,**rs**] [,**cs**[*n*]][,**ds**[*n*]][,**cd**[*n*]][,*data type*] [,**pe**][,**lf**]" [**for** *mode*] **as** [#]*buffer*[**len** = *number*]

Communications. Opens a file and assigns a buffer for RS-232C (asynchronous communications adapter) communication.

channel can be 1 or 2 to select the communications channel to open.

speed specifies the baud rate. It can be 75, 110, 150, 300, 600, 1200, 2400, 4800, or 9600. Default = 300.

parity can be e for even, o for odd, m for mark, s for space, or n for no. Default = e.

data specifies the number of bits. It can be 5, 6, 7, or 8. Default = 7.

stop can be either 1 or 2 to specify the number of stop bits. Default = 2 for baud rates of 75 and 100, and 1 for all other baud rates.

data type can be bin to transmit binary data (default) or asc for ASCII data.

mode is either output or input for sequential access. Default = random input/output.

buffer	specifies the buffer that accesses the file. It can be in the range 1-15.
number	specifies the maximum number of bytes that the **get** and **put** statements can access in the communications buffer. Default = 128 bytes.
rs	suppresses RTS (request to send).
cs[*n*]	controls CTS (clear to send).
ds[*n*]	controls DSR (data set ready).
cd[*n*]	controls CD (carrier detect).
lf	sends a line feed at each return.
pe	enables parity checking.
n	is the number of milliseconds to wait (0-65535) for that signal before a device timeout error occurs.

```
open "com1:" as 1
open "com1:9600,n,8,1,bin" as 2
```

option base *value*

Sets *value* as the minimum value for an array subscript. This statement must precede the **dim** statement.

value can be 1 or 0. Default = 0.

```
option base 1
```

out *port*, *data byte*

Sends a data byte to a machine output port.

port	is an integer in the range 0-65535.
data byte	is an integer in the range 0-255.

```
out 32,100
```

pcopy *source page, destination page*

Copies the *source* video page to the *destination* video page

```
pcopy 3,5
pcopy 6,4
```

peek(*memory location*)

Returns a byte from *memory location.*

memory location must be in the range -32767 to 65535. The value returned is an integer in the range 0-255.

```
a = peek(&h5a00)
b = peek(23040)
```

play *string*

Plays musical notes specified by *string.*

string is a string expression consisting of one or more of the following music commands:

Music Commands

a-g plays a note (A-G) on one musical scale. Include a number sign (#) or plus sign (+) to specify a sharp note. Include a minus sign (-) to specify a flat note.

l*n* sets the duration of the notes that follow. *n* can be a value in the range 1-64.

- 1 specifies a whole note.
- 2 specifies a half note.
- 4 specifies a quarter note.
- 8 specifies an eighth note.
- 16 specifies a sixteenth note.

o*n* sets the current octave (0-6). Octave 3 starts with middle C. Default = Octave 4.

n*n* plays a note. *n* can be in the range 0-84.

p*n* rests. *n* can be in the range 1-64.

t*n*	sets the number of quarter notes in 1 minute. *n* can be in the range of 32-255. Default = 120 quarter notes in 1 minute.
.	plays as a dotted note, increasing the value of the note by one-half.
mf	plays the music in the foreground. Default = mb.
mb	plays the music in the background. A maximum of 32 notes and/or rests at a time. Default = mb.
mn	sets *"music normal."* Each note plays 7/8 of the duration as set by the l option. Default = mn.
ml	sets "*music legato.*" Each note plays the full duration as set by the l option. Default = mn.
ms	sets "*music staccato.*" Each note plays 3/4 of the duration as set by the l option. Default = mn.
v*n*	sets the volume. *n* can be in the range 0-15 (default =8).
x*var*;	executes a substring represented by *var*. You can have one string execute another, which executes a third, and so on.

```
play "c4f.c8f8.c16f8.g16a2f2"
```

play(*number*)

Returns the number of notes in the background music queue. The maximum value play can return is 32 (the maximum buffer size).

number	is the voice channel (0, 1, or 2).

```
x=play(0)
x=play(2)
```

pmap(*coordinate*,*action*)

Returns the physical or world coordinate for the specified coordinate.

coordinate	is any x- or y-coordinate.
action	is one of the following:
0	returns the physical x-coordinate for the specified world *coordinate*.

1	returns the physical y-coordinate for the specified world *coordinate*.
2	returns the world x-coordinate for the specified physical *coordinate*.
3	returns the world y-coordinate for the specified physical *coordinate*.

```
x=pmap(200,0)
z=pmap(50,0)
```

point(*x, y*)
point(*action*)

Graphics. Returns the color number of a point on the screen, or returns the current physical or world coordinates.

(*x, y*)	are the coordinates of the point.
action	is one of the following:
0	returns the current physical x-coordinate.
1	returns the current physical y-coordinate.
2	returns the world x-coordinate if **window** is active. Otherwise, returns the physical x-coordinate.
3	returns the world y-coordinate if **window** is active. Otherwise, returns the physical y-coordinate.

```
10 if point(1,1)=0 then preset(1,1) else
   pset(1,1)
20 x=point(0)
```

poke *memory location*, *data byte*

Writes *data byte* into *memory location*. Both *memory location* and *data byte* must be integers and *memory location* must be in the range -32768 to 65535.

```
10 poke &h5A00, &hff
```

pos(*number*)

Returns the current column position of the cursor.

number is a dummy argument.

```
if pos(X)>70 then if a$ = chr$(32)
then a$ = chr$(13)
```

print ***data***[,***data***,...]

Prints numeric or string data on the display. If you use commas, the cursor automatically advances to the next tab position before printing the next item. If you use semicolons or spaces to separate the data items, **print** prints the items without spaces between them.

```
print"Do";"not";"leave";"spaces";
"between";"these";"words"
print "The total is",ttl
```

print using ***format***; ***data***[,***data***...]

Prints *data* using a format you specify.

format consists of one or more field specifier(s), or any alphanumeric character. Enclose *format* in quotation marks.

data can be a string, a numeric value, or both.

Specifiers for String Fields:

! prints only the first character in the string.

\spaces\ prints 2+*n* characters from the string.
(*n* is the number of spaces between the slashes.)

& prints the string without modifications.

Specifiers for Numeric Fields:

#	prints the same number of digit positions as number signs (#).
+	prints the sign of the number.
-	prints a negative sign after negative numbers (and a space after positive numbers).
**	fills leading spaces with asterisks.
$$	prints a dollar sign immediately before the number.
**$	fills leading spaces with asterisks, and prints a dollar sign immediately before a number.
,	prints a comma before every third digit to the left of the decimal point.
^^^^	prints in exponential format.
--	prints the next character as a literal character.

```
print using ".####^^"; 888888
print using "**$###,.##"; 1234.5
print using "####.#-"; -768.660
print using "###.##"; 876.567
```

print# ***buffer***,[**using** ***format***] ***data***[,***data***...]

Writes *data* items to a sequential access disk file. **print#** does not compress data before writing it to disk. It writes an ASCII-coded image of the data.

Refer to **print using** for information about the *format* parameter and specifiers.

```
print#1,a print#1, b$, t$
print#1,using "###.##";a(t)
```

pset [step](*x, y*)[,*color*]
preset [step](*x, y*)[,*color*]

Graphics. Draws a point on the display.

(*x, y*)	are the coordinates of the point.
step	appoints (*x, y*) as relative coordinates.

color specifies the color of the point. If you use **pset**, *color* defaults to the foreground color. If you use **preset**, *color* defaults to the background color.

```
pset (1,1)
preset step (1,1),0
```

put [#]*buffer*[,record]

Puts a record in a direct access disk file.

record is the record number to be written to the file and ust be in the range 1-16,777,215. Default = the current record number.

```
put 1
put 1,25
```

put [#]*buffer*, *number*

Communications. Transfers data from the communications buffer to the communications line.

number is the number of bytes to transfer.

```
put 2,80
```

put (*x*, *y*),*array*[,*action*]

Graphics. Transfers an image stored in an array onto the screen.

(*x*, *y*) are the coordinates at which the image begins (the upper left corner of the image). Default = the last point referenced.

array is the array variable name that holds the image.

action sets the type of interaction between the transferred image and the image already on the screen. *action* can be pset, preset, and, or, or xor. Default = pset.

```
put (200,100),a
```

randomize[*number*]

Reseeds the random number generator. If you omit the number parameter,GW-BASIC suspends program execution and prompts you for a number before executing.

number can be an integer or a single- or double-precision number.

```
randomize timer
randomize 300
```

read *variable*[,*variable*,...]

Reads values from a **data** statement, and assigns them to variables.

```
read t
read n$, d$, t
```

rem

Inserts a remark line in a program. You can use an apostrophe (') as an abbreviation for rem.

```
rem average velocity
'totals
```

renum [*new line*][,[*line*][,*increment*]]

Renumbers the program in memory including line numbers appearing after **goto, gosub, then, on/goto, on/gosub, on error goto, resume**, and **erl.**

line is the program line at which renumbering starts. Default = the first line.

new line is the new number assigned to *line*. Default = 10.

increment numbers the successive lines in the specified increments. Default = 10.

```
renum
renum 600, 5000, 100
```

reset

Closes all open files on all drives.

```
reset
```

restore [*line*]

Restores a program's access to previously read **data** statements.

line specifies the statement to access at the next read statement. Default = the first **data** statement.

```
restore
```

resume [*line* | **next**]

Resumes program execution after an error handling routine.

line branches to the specified line number. Default = the statement in which the error occurred.

next branches to the statement following the point at which the error occurred.

```
resume
resume 10
resume next
```

return [*line*]

Returns control from a subroutine executed by a **gosub** to the specified line. Default = the line immediately following the **gosub**.

```
return
return 40
```

right$(*string*,*number*)

Returns the specified number of characters from the far right portion of *string*.

number is an integer in the range 1-255.

```
print right$("watermelon",5)
print right$("puppylove",4)
```

rmdir *pathname*

Removes (deletes) the directory specified by *pathname*. This directory must be empty except for the "." and ".." symbols. See the MS-DOS **copy** and the BASIC **kill** commands.

```
rmdir "names"
rmdir "a:\accts\payable"
```

rnd[(*number*)]

Returns a random number in the range 0-1.

number — is an integer in the range -32767 to 32768. If *number* is negative, **rnd** starts the sequence of random numbers at the beginning. If *number* is 0, **rnd** repeats the last number generated. If *number* is a positive number, **rnd** returns the next number in the sequence.

```
print rnd(1)
a = rnd(0)
```

rset *field name* = *data*

Sets data in a direct-access buffer field name, in preparation for a **put** statement, and right-justifies it.

```
rset a$ = cvi(z)
```

run [*line*][**,r**]
run "*pathname*"[**,r**]

Executes the current program or loads and runs the specified program.

line — is the program line at which GW-BASIC begins execution. Default = the first line.

r — leaves the current files open when the new program, specified by *pathname* is loaded into memory.

```
run
run 100
run "program.a"
```

save "*pathname*" **[,a|,p]**

Saves a program on disk with the specified name. If you do not specify a format, **save** uses the compressed format.

a	saves the program in ASCII format.
p	saves the program in a protected format.

```
save "a:file.1.bas"
save "\educ\mathpak.txt",a
```

screen (*row*,*column*[,*number*])

Returns the ASCII code for the character at the specified *row* and *column*. If *number* is specified and is non-zero, GW-BASIC returns the color attribute.

```
a = screen(20,20)
print screen(10,10,1)
```

screen [*mode*][,[*burst*][,[*active page*][,[*display page*][,*erase*]]]]

Sets the screen mode and screen attributes to be used by all other graphics statements.

mode	is an integer in the range 0-2.
burst	activates or de-activates *color* in Screen Mode 0 or 1. *burst* has no effect on Screen Mode 2. In Mode 0, use 1 to activate, 0 to de-activate. In Mode 1, use 0 to activate and 1 to de-activate.
active page	selects the video page to which GW-BASIC is to write. Default = the current active page.
display page	is an integer that selects the video page GW-BASIC is to display. Default = the current active page.

erase indicates how much video memory to erase. *Erase* can be:

0	do not erase memory, even if the screen mode changes.
1	erase the union of the new and old pages if *mode* or *burst* changes (default).
2	erase all video memory if *mode* or *burst* changes.

```
screen 0,0
screen 2
screen 8,1
```

sgn(*number*)

Determines *number*'s sign.

If *number* is negative, **sgn** returns -1. If *number* is positive, **sgn** returns 1. If *number* is 0, **sgn** returns 0.

```
y = sgn(a * b)
```

shell(*command*)

Loads and executes another program (a program with either a .exe or .com extension) or an internal command as a child process to the original program.

command is a string expression containing the name of the program you want to run.

```
shell "format b:"
```

sin(*number*)

Returns the sine of *number*.

number must be in radians.

```
print sin(7.96)
d=sin(t)
```

sound *frequency*,*duration*[,*voice*] | **on** | **off**

Generates a sound with the frequency and duration specified.

frequency	is an integer in the range 1-1023, indicating the tone in hertz.
duration	is a numeric expression in the range 1-65535, specifying the duration in clock ticks.
voice	identifies the defined sound as voice 0, 1, or 2.
on	enables the external speaker.
off	disables the external speaker.

```
sound 37,2
```

space$(*number*)

Returns a string of *number* spaces.

```
print "Cost" space$(4)
"Quantity" space$(9) "Total"
```

spc(*number*)

Skips *number* spaces in a **print** statement.

```
print "Hello" spc(15) "there"
```

sqr(*number*)

Returns the square root of *number.*

number	must be greater than zero.

```
print sqr(155.7)
```

stop

Stops program execution.

```
stop
```

str$(*number*)

Converts *number* to a string.

```
s$ = str$(x)
print str$(-234)
```

string$(*number*,*character*)

Returns a string containing the specified number of characters.

number must be in the range 0-255.

character is a string or an ASCII code.

```
b$ = string$(25, "x")
print string$(50, 10)
```

swap(*variable1*,*variable2*)

Exchanges the values of two variables of the same type.

```
swap f1#,f2#
```

system

Returns you to the MS-DOS command level.

```
system
```

tab(*number*)

Spaces to position *number* on the display.

number must be in the range 1-255.

```
print "Name" tab(25) "Amount":print
```

tan(*number*)

Returns the tangent of *number*.

number must be in radians.

```
print tan(7.96)
s = tan(x)
```

time$[=*string*]

Sets the time to the specified string or retrieves the current time if you omit *string*. GW-BASIC uses a 24-hour clock.

string is a literal, enclosed in quotation marks.

```
time$ = "14:15"
print time$
```

timer

Returns the *number* of seconds since midnight or since the last system reset.

```
a = timer
print timer
```

timer *action*

Turns on, turns off, or temporarily halts timer event trapping.

action can be on, off, or stop.

```
timer on
timer off
timer stop
```

troff
tron

Turns on (tron) or turns off (troff) the program flow tracer.

```
tron
troff
```

usr[*number*](*argument*)

Calls the user's assembly-language subroutine identified with *number*, and passes *argument* to that subroutine. The number you specify must be the same as the corresponding **def usr** statement for that routine. Default = 0.

number specifies a number in the range 0-9 which identifies the subroutine being called as defined with the **def usr** statement.

argument is a numeric or string expression passed to the subroutine.

```
usr("100")
```

val(*string***)**

Calculates the numerical value of *string*.

string is a numeric string.

```
print val("100")
print val(num$)
```

varptr(*variable***)**
varptr([#]*buffer***)**

Returns the offset into GW-BASIC's data segment of a variable of the file control block.

When used with *variable*, **varptr** returns the address of the first byte of data identified with *variable*.

variable is the name of the variable for which you want the address.

When used with *buffer*, it returns the address of the file's control block

buffer is the number assigned to the file when it was opened.

```
a = varptr(a$)
print varptr(3)
```

varptr$(*variable***)**

Returns a 3-byte string representing the memory address of variable. Byte 0 is the type, Byte 1 is the low byte of address, and Byte 2 is the high byte of address. Type 2 is for integer variables, 3 for string variables, 4 for single-precision variables, and 8 for double-precision variables.

variable is the name of the variable for which you want the offset.

```
play "x" + varptr$(a$)
```

view [screen] [(*x1*, *y1*)-(*x2*, *y2*)[,[*color*][,[*border*]]]]

Graphics. Creates a viewport that redefines the screen parameters.

(*x1*, *y1*)	specifies the upper-left coordinates for the rectangular viewport.
(*x2*, *y2*)	specifies the lower-right coordinates for the rectangular viewport.
color	the color with which to fill the viewport.
border	an integer specifying the color for the boundary line around the viewport.
screen	specifies that all coordinates used in drawing are absolute to point 0,0 on the screen. If you omit **screen**, all coordinates specified are relative to the viewport coordinates.

```
view (10,10)-(100,100)
view screen (20,25)-(100,150)
```

view print *top line* to *bottom line*

Creates a text viewport that redefines the text screen parameters.

top line	specifies the first line of the text viewport. It can be in the range 1-24, but must be less than *bottom line*. Default = Line 1.
bottom line	specifies the last line of the text viewport. It can be in the range 1-24, but must be greater than *top line*. Default = Line 24.

```
view 1 to 15
```

wait *port*, *number1* [,*number2*]

Suspends program execution until the specified machine input port develops a specified bit pattern.

number1	an integer in the range 0-255.
number2	an integer in the range 0-255.

```
wait 32,2
```

while *expression* **wend**

Executes a series of statements in a loop as long as a given condition is true.

expression — is a logical expression that returns either true or false. If e*xpression* is true, GW-BASIC executes the statements after the **while** statement until it encounters a **wend** statement. If *expression* is still true, GW-BASIC repeats the process. If it is not true, execution resumes with the statement following the **wend** statement.

```
while a
print "Calculating..."
wend
```

width [lprint] *size*
width *buffer*, *size*
width *device*, *size*

Sets the line width in number of characters for the display, printer, or communications channel.

size — an integer in the range 0-255 that specifies the number of characters in a line. For the screen, *size* can be only 40 or 80. Default = 255 (communications channel).

buffer — is the number assigned to the file in the **open** statement.

device — is a valid device, enclosed in quotation marks, that specifies the device for which you are setting the width. It can be scrn:, lpt1:, com1:, or com2:.

```
width 40
width lprint 100
width "scrn:",40
```

window [screen] [(*x1*, *y1*)-(*x2*, *y2*)]

Lets you change the physical coordinates of the screen (or current viewport). Lets you plot points outside the normal screen coordinate limits by setting new world coordinates to the screen.

(*x1*, *y1*)	are the world coordinates for the upper-left corner of the screen.
(*x2*, *y2*)	are the world coordinates for the lower-right corner of the screen.
screen	tells GW-BASIC to set the coordinates like the screen display. If you omit **screen,** GW-BASIC inverts the y-coordinates to show a true Cartesian coordinate system.

```
window (1984,100000)-(1987,300000)
```

write *data*[,*data*...]

Outputs *data* to the screen.

```
10 A=80:B=90:c$="That's All"
20 write a,b,c$
```

write# *buffer*, *data*[,*data*...]

Writes *data* to a sequential-access disk file.

```
write#1,A$,B$
```

GW-BASIC Function Key Settings

F1	list"	**F6**	,"lpt1:" **ENTER**
F2	run **ENTER**	**F7**	tron **ENTER**
F3	load"	**F8**	troff **ENTER**
F4	save"	**F9**	key
F5	**CONT ENTER**	**F10**	screen 0,0,0 **ENTER**

Typing Keywords Using the ALT Key

ALT-A	auto	**ALT-N**	next
ALT-B	bsave	**ALT-O**	open
ALT-C	color	**ALT-P**	print
ALT-D	delete	**ALT-Q**	(none)
ALT-E	else	**ALT-R**	run
ALT-F	for	**ALT-S**	screen
ALT-G	goto	**ALT-T**	then
ALT-H	hex$	**ALT-U**	using
ALT-I	input	**ALT-V**	val
ALT-J	(none)	**ALT-W**	width
ALT-K	key	**ALT-X**	xor
ALT-L	locate	**ALT-Y**	(none)
ALT-M	motor †	**ALT-Z**	(none)

† **Motor** is a reserved word, but it is not recognized in this implementation of GW-BASIC.

Exponential Notation and Numeric Precision Characters

D	Used in double-precision exponential notation.
E	Used in single-precision exponential notation.
%	Makes the variable preceding it integer-precision.
!	Makes the variable preceding it single-precision.
#	Makes the variable preceding it double-precision.
$	Makes the variable preceding it a string.

Operator Precedence

Each operator or group of operators takes precedence over the group below it.

( )	Parentheses
^	Exponentiation
+ –	Unary positive, negative
* /	Multiplication, division
\	Integer division
MOD	Modulus arithmetic
+ –	Addition, subtraction
< > <=	Relational tests
>= <>	
NOT	
AND	
OR	
XOR	
EQV	
IMP	

Video Modes

Screen Mode 0

Graphics = No

Resolution = N.A.

Colors = 16

Text Width = 40/80

Maximum Pages = 8 at Width 40, 4 at Width 80

Video Page Size = 2048 at Width 40, 4096 at Width 80

Screen Mode 1

Graphics = Yes

Resolution = 320 x 200.

Colors = 4 colors 2 palettes

Text Width = 40

Maximum Pages = 8

Video Page Size = 16384

Screen Mode 2

Graphics = Yes

Resolution = 640 x 200

Colors = monochrome

Text Width = 80

Maximum Pages = 1

Video Page Size = 16384

Error Codes and Messages

Number	Message
1	NEXT without FOR
2	Syntax error
3	RETURN without GOSUB
4	Out of DATA
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Redimensioned array/duplicate definition
11	Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No RESUME
20	RESUME without error
21	Unprintable error
22	Missing operand
23	Line buffer overflow
24	Device timeout
25	Device fault
26	FOR without NEXT
27	Out of paper

Number	Message
29	WHILE without WEND
30	WEND without WHILE
50	FIELD overflow
51	Internal error
52	Bad file number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
61	Disk full
62	Input past end
63	Bad record number
64	Bad file name
66	Direct statement in file
67	Too many files
68	Device unavailable
69	Communication buffer overflow
70	Disk write protect
71	Disk not ready
72	Disk media error
73	Advanced feature
74	Rename across disks
75	Path/File access error
76	Path not found
77	Deadlock
78	Unprintable error

RADIO SHACK
A Division of Tandy Corporation
Fort Worth, Texas 76102

12A8

Printed in Japan
2EC6P1MA09100

ADDENDUM
TANDY 1400 FD/HD
Cat. no. 25-3501/3505

Note: Contrary to any information in the *GW-BASIC Quick Reference Guide*, you can only access communications channel 1 using a BASIC program. You cannot access communications channel 2 from BASIC.

Radio Shack
Fort Worth, TX 76102